To run the extension, see line 8 on server.py for where to type in the API key.

Video Demonstration:

The overall goal of our project is to provide an on-the-spot, interactive tool that integrates AI tutoring into any Chrome-based coding session. This allows users to practice and reinforce their understanding of code in real-time, without needing to leave their work environment or disrupt their coding flow. Our primary target audience is individuals who are new to coding or are looking to learn a new programming language. The extension is also useful for more experienced coders, who can use it for tasks such as optimizing syntax or gaining a deeper understanding of others' code. While a simple overview and demonstration of the extension's functionality has been provided in the above video, we also want to offer a more detailed summary of the extension's key features and expand upon the core aspects of the project.

One of the standout features of the extension is the pop-up window that houses the user interface. The user can select lines of text on any website in Google Chrome, but we specifically experimented with Google Colab and Stack Overflow, as these are both widely used by developers and offer high-quality coding resources. The extension operates as a browser extension, meaning it must be enabled before use. Once the user highlights a section of code, a round button with the label "Explain Now" will appear. When the user clicks this button, the selected code is sent directly to GPT-4o Mini, which provides an explanation of the code in the pop-up interface. This window also includes an input field where users can ask follow-up questions or upload supplemental code for more detailed information.

Implementing the extension was a challenging process because many of us had no prior experience writing browser extensions. A significant portion of the development, especially for the CSS and HTML elements of the user interface, was assisted by ChatGPT. While ChatGPT provided us with a basic framework and code, we made extensive edits to improve the design, enhance the user experience, and make the interface more sleek and visually inviting.

One of the more technical challenges we faced was integrating the API key and setting up the necessary infrastructure to run the extension. Specifically, the need to install the libraries of OpenAi, Flask, and Flask-Cores as well as execute the "python server.py" command in the terminal within the project folder posed a barrier to ease of use. At this stage, the extension cannot yet function independently within Google Chrome, as it requires running a local server on the user's machine. This means users must enable developer mode on their local device and

perform additional setup steps. While this adds some complexity to the initial installation process, it is an area we are actively working to streamline. A future improvement could involve using cloud servers, such as AWS, to host the server-side code, which would eliminate the need for users to manage a local server and make the extension more accessible.

Another limitation we're aware of is that our extension currently only supports Google Chrome, which means it is not usable for individuals who either cannot access or prefer not to use Chrome. Additionally, coders who work offline, using desktop IDEs or other development environments, are unable to benefit from the extension. Extending the functionality to other browsers or offering compatibility with offline IDEs would significantly increase the extension's versatility and reach. We have already begun exploring how this could be implemented in future iterations of the extension.

The actual prompting mechanism of the extension, which explains the highlighted code, was initially planned to also include practice problems for users to solve. However, our early versions of the user interface were size-locked, and couldn't display larger responses for more complex code. We found that even simple explanations could be cut off if the response exceeded the available space. Furthermore, generating practice problems proved to be a challenge, particularly when trying to identify which sections of syntax users were struggling with in longer code snippets. To address this, we significantly expanded the user interface, adding features such as a drop-down explanation option, a scroll bar, and improved scroll functionality. These changes enabled us to display longer and more detailed responses, improving the overall experience for users. Additionally, we incorporated an input bar for users to ask follow-up questions, further enhancing the interactive learning aspect.

While our initial extension only worked well for isolated selections of code, we implemented a key feature: the ability to contextualize the explanations based on a larger code snippet. This is a complex challenge because it requires the extension to retrieve and analyze code that isn't highlighted. The difficulty lies in passing this additional context to GPT-4o Mini in an efficient and effective way, while ensuring the responses are relevant to the larger code structure. Now, users can upload the additional code as a file not larger than 1 MB in the same area as the follow-up questions and press return for it to be sent to the AI tutor. Through this step, the AI tutor is able to offer more nuanced and comprehensive explanations, enhancing the tool's utility for users working with larger or more complex projects.